Amendments to Claims

This listing of claims will replace all prior versions and listings of claims in the application:

Listing of Claims

1. - 9. (canceled)

10. (currently amended) A method of deadlock management in a multi-thread, parallel processing data management system having ports for sending and receiving data tokens comprising:

allocating at least one thread to a first process and at least one thread to a second process, wherein the first and second processes are connected through a queue via sending and receiving ports;

determining if a thread is blocked, waiting on another thread, and determining if the blocked thread is sending data or receiving data, wherein a receiving port blocks if a data token is unavailable and a sending port blocks when a queue limit is reached; and

determining if a deadlock exists by building a wait graph of blocked threads in the system and determining if the graph is cyclic, that is waiting on itself, indicating a deadlock does exist.

11. (original) The method of claim 10, blocking a receiving port when a data token is not available.

12. (original) The method of claim 10, blocking a sending port when a limit on the number of data tokens in the queue is reached.

13. (original) The method of claim 10, including building a wait graph with said blocked threads and traversing said wait graph to determine if it is cyclic.

14. (original) The method of claim 10, if a deadlock is detected, correcting the deadlock by allowing the limit of data tokens on a queue to increase.

15. (original) The method of claim 14, wherein the limit of a queue associated with a sending port is allowed to increase.

16. (original) The method of claim 14, wherein the token batch size of another queue is decreased while said limit of said queue is increasing.

17. - 20. (canceled)

21. (new) A method for executing a dataflow application comprising:

providing a dataflow application comprising a plurality of map components and data ports, a number of map components being linked between data ports and each map component some map components comprising one or more composite components having a plurality of processes and at least some of said linked data ports having a queue;

allocating a processing thread to a respective process composite map component;

executing multiple processes in parallel with each composite map component on a separate thread;

detecting if a deadlock condition does or will exist for a thread by building a wait graph of several thread states and determining if the wait graph is circular; and

correcting a deadlock by allowing a queue linking data ports to exceed a queue limit.

22. (new) The method of claim 21, wherein the correcting step includes choosing a thread that waits as a producer if a circular wait graph is detected.

23. (new) The method of claim 21, wherein if the detecting step determines a wait graph is circular, the correcting step including analyzing queues other than the allowed queue in the wait graph for token batch reduction.

24. (new) The method of claim 21 wherein a wait graph is circular, while allowing a queue to exceed a queue limit, the substep of reducing toke batching in other queues in the wait graph.